陰関数描画と区間数演算の効率化について (Computer Algebra : Design of Algorithms, Implementations and Applications)

AUTHOR(S):

村尾, 裕一; 齋藤, 友克; 近藤, 祐史

CITATION:

村尾, 裕一 ...[et al]. 陰関数描画と区間数演算の効率化について (Computer Algebra : Design of Algorithms, Implementations and Applications). 数理解析研究所講究録 2009, 1652: 112-119

ISSUE DATE:

2009-06

URL:

http://hdl.handle.net/2433/140808

数理解析研究所講究録
第1652巻 2009年 112-119

陰関数描画と区間数演算の効率化について

村尾 裕一
電気通信大学・電気通信学部*
HIROKAZU MURAO
UNIV. ELECTRO-COMMUNICATIONS

齋藤 友克
アルファオメガ†
TOMOKATSU SAITO
ALPHAOMEGA INC.

近藤 祐史
徳山工業高等専門学校‡§
YUJI KONDOH
TOKUYAMA NATIONAL COLLEGE OF TECHNOLOGY

1 はじめに – 経緯とあらまし

数式処理システム Risa/Asir には，陰関数をも正確に描画するための独自の方法 [5, 2] が実装され（関数 ifplot），特徴的な機能のひとつとなっている．しかしその実装法はやや旧式であるため，より現代的なものとすべく描画アルゴリズムから実装法までの広範囲な見直しを行い，改良と開発を進めている．[3] では，領域表示と3次元表示の試みという機能強化を行うと共に，曲線描画を描画空間全体からの描画領域（セル）の探索と捉えてアルゴリズムの改良を行った．Ifplot では描画領域を覆うすべてのセルについて，セルが表す空間（点の集合）中に描画する式を満たす点が存在するかを判定する．その判定法としては何種類か提案されているが，そのひとつとしてセルを区間数で表し，描画関数をその区間数で評価して関数を満たす可能性のない領域は除外していくという方法がある．改良されたアルゴリズムでは，この探索問題に対し，領域を半分ずつに狭めていくという二分法を適用し高速化を図った．また，我々の手法では空間中のすべてのセルに対し評価を行うことが基本なので相当な計算量を要する可能性があるし，その評価は数式の数値化という数値的処理であるので数式処理の計算としては珍しい，並列処理に適合し易く適合すべき処理である．我々は，今日 PC の CPU で一般的になってきているマルチコアの CPU を用いて並列処理を試み，十分な効果のあることを確認した [4]．マルチコア並列処理については，汎用ではなくグラフィックス用の GPU においてはるかに高性能なものが流通するようになり，これを一般の数値処理に利用するための研究が盛んに行われている．

我々は今回の描画機能の改良と機能拡張に当たり，こうしたマルチコア CPU でのメモリ共有のマルチスレッド並列処理も視野に入れて，新たに実装することを計画している．本稿では，そのための計算手法を確立すべく，従来の計算法を見直しや経験則の正当性を検討してより良い方法を探る．本稿で用いるアルゴリズムは，区間数を用いた上述の探索法である．

*murao at cs.uec.ac.jp
†saito at a2z.co.jp
‡y-kondou at tokuyama.ac.jp
§1年間の高専間交流期間中．2008年4月以降は詫間電波高専に復帰．

2 描画と指標関数

Risa/Asir の描画関数 ifplot の描画法について説明する．描画空間は有限の大きさを持つセル（３次元ならばボクセル）で覆いつくされると考え，各描画セルは空間中の微小な矩形領域を表すとみなし，すべてのセルに対し表示するか否かの判定を行う．即ち，物理的に大きさを持つ描画点は本来は大きさを持たない数学的な意味での点の集合とみなし，この点において描画する点がある空間中の一点を表すとする一般的な方法と大きく異なる．描画は，基本的に，セル中に解が存在する場合に点灯するという方針で行うが，空間中のどこに解があるかを知ることは一般には難しく，(特に曲線を追跡しながら描いていく場合には）容易に見落とすことになるので注意が必要である．

- 曲線描画とは ... 描画空間中での描画点の探索である．
- ある（矩形）領域中に描画関数の零点が存在しうるかの評価関数 ..... 指標関数（character func.）
- （最初は全表示空間に零点は存在しうると仮定し）存在しないと判定できた領域は消していく
- 指標関数 χ に対する 強い要請：領域 C に対し，$\chi(C)=0$ ならば $\forall x \in C. f(x) \neq 0$.

指標関数が強い要請を満す時 faithful であると呼んでいる．

色々な指標関数　これまでに，上記の基本方針とは異なるものも含め，数種類の指標関数が提案されている．

- **interval** char.（区間指標関数）：描画セルに対応する区間数で描画関数を評価し，評価値の区間数に 0 が含まれるなければ零点は存在しないと判定．描画関数の形によっては（多項式など） faithful（強い要請を満たす）である．
- **boundary** char.: セルの境界は直線とし，その境界の線分上に解があるかを Sturm 列により判定．
- **sign** (weak) char.: 中間値の定理を根拠として，セルの四隅での関数の評価値の符号が皆同じでなければ零点が存在すると判定 .....(ifplot の標準)

後の２つは要請を実用上問題がない程度に緩和したもので，描画したセル中に零点が存在することを保証するものである（要請「選択されないセル中に実零点は存在しない」の保証まではない）．

以上が ifplot の開発当初から用いられてきたものだが，最近我々は ifplot の機能強化や並列処理の導入も考慮に入れた効率の改善を目指して改良を開始し，同時に複合指標関数という新たな手法を提案している．

- 複合指標関数: interval char. と sign weak char. の両方を使う．即ち，多項式の描画関数に対しては interval char. が faithful であることを生かし，解の存在しない領域をより大きなセルとして除外することを試みる．即ち，領域全体を 2×2 の４つの領域に分割し，それぞれに対し interval char. を用いて零点の存在を判定し，零点が存在する可能性がある領域に対してはこの判定を再帰的に適用する．しかし，interval char. では，区間数の計算に伴う区間の膨張があるために，零点を含まないセルでも含まないと断定することは難しい（より微細なセルとしての計算が必要）．我々は，現実的な対応をするために，大きさが十分に小さくなったセルに対しては sign weak char. を用いることとする．

3 多項式評価の並列処理

我々は描画機能の強化と改良の一環として実装法の見直しも進めており，今日一般的となったマルチコア CPU を活用し並列処理を積極的に導入することを計画している．ここでは，今後の検討材料とするために，これまでの実験内容と結果のまとめを行う．

- 計算の対象は，sign char. 指標関数で用いる多項式の値の評価（倍精度浮動小数として）
- 並列処理中の（構造データの生成に必要となる）メモリ管理を避けるべく，評価法の簡略化[1]と，配列の利用をすすめた（中間結果の多項式の配列による表現．中間結果の配列用のメモリ領域は予め確保．ベクトル演算の導入）．
  - （専用のデータ構造：一変数多項式）Horner 法向けの配列表現多項式：

    $$x^{e_n}(x^{e_{n-1}}(\cdots(x^{e_1}c_1+c_2)\cdots)+c_{n-1})+c_n$$

    項数 n，$\{e_j\}$ の配列及び $\{c_j\}$ の配列で表現．$x=(x_1,...,)$ で評価する関数．
  - 二変数多項式は，上と同じ形で係数を上の形の一変数多項式とする．一方の変数への値の代入は上の一変数多項式を生成することに注意．
- 並列処理の記述は OpenMP でお手軽に（for ループの単純な分割する work-sharing）．但し，粒度ができるだけ大きくなるように計算順序を計画する．
- 【実験計測結果】画素数（セルの個数）が多いほど，あるいは，D,H,S,C と式が複雑になるほど並列処理の効果が大きくなる．2× デュアルコア CPU の 4CPU の利用で 2.4 倍まで観測した．

区間指標関数を用いるには多項式を区間数（両端は double 型）で評価することになるが，同様の方法が実現可能であろう．

4 区間数での多項式の評価

区間数の演算

ここでは区間数での多項式の値を評価する方法を検討する．そのために先ず，区間数の算術演算をまとめその性質を明らかにする．

- 加算：$[l_1,h_1]+[l_2,h_2]=[l_1+l_2,h_1+h_2]$．$[l,h]+c=[l,h]+[c,c]=[l+c,h+c]$，
- 減算：$[l_1,h_1]-[l_2,h_2]=[l_1-h_2,h_1-l_2]$．よって，符号反転は $-[l,h]=[0,0]-[l,h]=[-h,-l]$，
- スカラー倍：$c[l,h]=\begin{cases}[cl,ch], & c>0 \text{ の時}\\ [ch,cl], & c<0 \text{ の時}\end{cases}$
- 区間数どうしの乗算：

	$[l_1,h_1\le 0]$	$[l_1\le 0,0<h_1]$	$[0<l_1,h_1]$
$[l_2,h_2\le 0]$	$[h_1h_2,l_1l_2]$	$[h_1l_2,l_1l_2]$	$[h_1l_2,l_1h_2]$
$[l_2\le 0,0<h_2]$	$[l_1h_2,l_1l_2]$	$[\min(l_1h_2,h_1l_2),\max(l_1l_2,h_1h_2)]$	$[h_1l_2,h_1h_2]$
$[0<l_2,h_2]$	$[l_1h_2,h_1l_2]$	$[l_1h_2,h_1h_2]$	$[l_1l_2,h_1h_2]$

- 冪乗：$[l,h]^e$（e は正整数とする）．$[l,h]^e=\begin{cases}[l^e,h^e], & e \text{ が奇数，または，} 0\le l\le h\\ [0,h^e], & l\le 0\le h \text{ かつ } |l|\le h\\ [0,l^e], & l\le 0\le h \text{ かつ } h\le|l|\\ [h^e,l^e], & l\le h\le 0.\end{cases}$

[1] 実はこの書き換えによる無駄なリスト処理の効果が絶大で，大幅な高速化を実現している．

- 逆数：$1/[l,h] = \begin{cases} [1/h, 1/l], & 0 \notin [l,h] \text{ の時} \\ [-\infty, \infty], & l \leq 0 \leq h \text{ の時} \end{cases}$

形式的あるいは機械的には，演算結果は上のようになるが，(正確な値 ± 誤差) という区間数が表す本来の意味が乗算や冪乗の結果では把握し難くなっている．後で説明するように，上下限の値 l_s と h_t が不自然に混ざってしまったり上下限の位置が変わってしまう場合，計算精度の悪化を招く可能性がある．このことは区間 $[l,h]$ に 0 が含まれる時に顕著であり，実際，冪乗 $[l,h]^2$ と積 $[l,h][l,h]$ では結果が異なる．より詳しくは [1] などを参照．

区間数での多項式の評価法の検討

次に，多項式の区間数での評価について検討する．我々は経験的に次のことが成り立つと推測した．

- 評価位置 (X や Y の値) を，上限か下限が 0 となるようにシフトすると精度が良くなる．即ち，$f(X)$ を $X = [a, a+\alpha]$ で評価する時，$g(X) = f(X+a)$ を計算した後 $g([0,\alpha])$ を評価する方が精度が良い (数式処理的手法が必要)．
- 多項式の評価にはホーナー法を用いると (項毎に計算して足し合わせるより) 精度が良い．

ここで「精度が良い」とは「評価結果の区間数の幅がより狭い」を意味する．後者は，区間演算の準分配則 $(I_1+I_2)I_3 \subseteq I_1I_3 + I_2I_3$ に根拠があると思われる．これらの経験則と正当性をより詳しく調べてみよう．

平野は [6] において次の事実を示している．

> 1 変数多項式をホーナー法で評価する場合，シフトは有効，即ち，評価結果の区間の幅がシフトしない場合より拡がる事はない．

区間演算による多項式の評価では，上記の準分配則の観点から「ホーナー法が良い結果を与える」とし，ホーナー法を定式化して 1 ステップ毎の値の範囲の変化を追跡することで，一端を 0 にシフトすると評価値の幅が拡がることはない事を示し，より良い結果を与える場合が多いとしている．ここでの議論は 2 変数の場合についても，係数が区間数の多項式を扱うと考えれば，最良とは言えなくとも同程度の議論 (範囲を押さえること) が可能であろう．さて，ここで示された事実は上の経験則の各々に対する答とはならないし，また，常に良い結果になるのか，どのような条件の時に良い結果になるのか，他に良い方法はないのか (つまり，この方法が最善なのか) といった疑問には答を与えていない．特に，シフトは式の計算が必要なばかりか一般に式の膨張を招くので，より良い結果となる確率が低いのであれば使いたくはない手法である．また，ホーナー法が良い計算法であることは事実としても，更によくする方法が考えられるかもしれない．

例として $f(X) = (X+A)^n = X^n + {}_nC_1X^{n-1}A + \cdots + A^n$ という単純な多項式を，$[a, a+\alpha]$ で評価することを考えてみよう．$f_1(X) = X + {}_nC_1A$ 及び $f_k(X) = f_{k-1}(X)X + {}_nC_kA^k$ とおく．ホーナー法による $f([a,a+\alpha])$ の評価は，

(1) 初期値：$f_1([a,a+\alpha]) = [a + {}_nC_1A, a+\alpha+{}_nC_1A]$.

(2) $f_k([a,a+\alpha]) = f_{k-1}([a,a+\alpha])\,[a,a+\alpha] + {}_nC_kA^k$ の計算を繰り返し，

(3) 最終的に $f([a,a+\alpha]) = f_n([a,a+\alpha])$.

シフトした場合 $g(X) = f(X+a) = (X+a+A)^n = X^n + \cdots + (a+A)^n$ も同様に，$g_1(X) = X + {}_nC_1(a+A)$ 及び $g_k(X) = f_{k-1}(X)X + {}_nC_k(a+A)^k$ とおく．ホーナー法による $g([0,\alpha])$ の評価は，$g_1([0,\alpha]) = [{}_nC_1(a+A), \alpha + {}_nC_1(a+A)]$ とし，$g_k([0,\alpha]) = g_{k-1}([0,\alpha])\,[0,\alpha] + {}_nC_k(a+A)^k$ の

計算を繰り返し，最終的に $g([0,\alpha]) = g_n([0,\alpha]) = f([a,a+\alpha])$ を得る．精度の悪化の問題は区間数どうしの乗算 $f_{k-1}([a,a+\alpha])[a,a+\alpha]$ 及び $g_{k-1}([0,\alpha])[0,\alpha]$ で，$f_{k-1}()$ や $g_{k-1}()$ の値の区間数が 0 を含む時に起こりうる．逆に，$A > 0$ かつ $a > 0$（十分条件）の場合のように $f_k([a,a+\alpha]) = [\sum_{j=0}^{k} {}_nC_j A^j a^{k-j}, \sum_{j=0}^{k} {}_nC_j A^j (a+\alpha)^{k-j}]$ 及び $g_k([0,\alpha]) = [{}_nC_k (a+A)^k, \sum_{j=0}^{k} {}_nC_j (a+A)^j \alpha^{k-j}]$ の符号が一定であれば，$f([a,a+\alpha]) = g([0,\alpha]) = [(a+A)^n, (a+A+\alpha)^n]$ と一致するのでシフトすることは無駄である．同じように $|a| \gg 1$（A との比較で）ならば上と同様の議論が可能なので，結果はシフトに依存せずシフトは不要である．これは，$f(X) = 0$ の根からは大きく離れた単調増加（あるいは減少）になった領域での評価となり，計算途中での区間数も 0 を含まないためである．

ここで，一般に関数が単調であることは大変有用であることを指摘しておく．関数が連続で単調であれば，次が成り立つ．

$f(X)$ が $[l,h]$ の範囲で単調増加（または減少）ならば $f([l,h]) = [f(l), f(h)]$（または $[f(h), f(l)]$）．

区間が拡張し精度が悪くなるのは主に 0 を含む区間数の積を計算した時に本来は無関係の区間の上下限の値の積を計算する場合である．関数が単調とわかっていれば，上式のとおりで，そのような状況を招かずに済ませられる．ここでホーナー法の 1 ステップに相当する $f(X) = ([a_l, a_h]X^d - [b_l, b_h])X^e$ の $X = [l,h]$ での評価を考えてみよう．簡単のため区間数の範囲は正とする（よって 0 を含まない）．

- 【項の和】$[a_l, a_h][l^{d+e}, h^{d+e}] - [b_l, b_h][l^e, h^e] = [a_l l^{d+e} - b_h h^e, a_h h^{d+e} - b_l l^e]$,
- 【ホーナー法】$([a_l, a_h][l^d, h^d] - [b_l, b_h])[l^e, h^e] = [a_l l^d - b_h, a_h h^d - b_l][l^e, h^e]$.

ホーナー法による値は，$[a_l l^d - b_h, a_h h^d - b_l]$ の両端の値が共に負/符号が異なる/共に正により $[a_l l^d h^e - b_h h^e, a_h h^d l^e - b_l l^e]$, $[a_l l^d h^e - b_h h^e, a_h h^{d+e} - b_l h^e]$, $[a_l l^{d+e} - b_h l^e, a_h h^{d+e} - b_l h^e]$ となり，いずれの場合も項の和による計算よりも幅が狭くなることがわかる．共に正の場合の上下限値は $X = h, l$ での値に一致し l と h が混ざった式にはならないが，$f(X)$ が単調増加な領域での上下限値として簡単に得られる．即ち，関数形により上下限値が容易にわかる場合は（当然のことながら）区間数の一般的な計算法を用いる必要はない．その特別な場合として単調な場合があげられるが，関数の傾きがある領域内で 0 となる可能性があるかの判定は 2 次式ならば容易に行える．このことを発展させれば，ホーナー法を通常 1 項ずつ加えて行く計算を単調と判定できる場合には 2 項（2 次式）ずつ加えていくことにより改良可能と考えられる．こうした細かな改良法の検討とアルゴリズムの開発は今後の課題である．

5 四分木アルゴリズムの改善

四分木 (クォドツリー : quad-tree) に基づく描画アルゴリズムとは，次の処理を再帰的に適用する方法である．

- 描画領域 $([l_x, h_x], [l_y, h_y])$ 内に零点が存在しないと判定されれば（$0 \notin f([l_x, h_x], [l_y, h_y])$），何も描画せずに終了．
- 領域が十分に小さければ，再帰せずに然るべき判定を行う（指標関数では零点の存在を否定できないとして点灯（描画）するか，別の指標関数を用いて描画を行う）．
- さもなくば $m_s = (l_s + h_s)/2$, $s = x, y$ として，領域を次のように 4 分割し

  $([l_x, m_x], [m_y, h_y])$, $([m_x, h_x], [m_y, h_y])$,
  $([l_x, m_x], [l_y, m_y])$, $([m_x, h_x], [l_y, m_y])$.

  判定を再帰的に繰り返す．

シフト計算の効率化

式の評価にホーナー法を用いるのであれば区間の一端を 0 にシフトすべきだが，これは式中の変数への別の式の代入という数式処理の操作であり，それなりのコストを要する．このシフトのための式の計算の回数は，4 分割と同時に原点を (m_x, m_y) にずらした式 $(g(X,Y) = f(X+m_x, Y+m_y))$ を計算し次の 4 つの領域に共通に用いることで減らすことができ，効率の改善につながる．

$$([l_x - m_x, 0], [0, h_y - m_y]), \quad ([0, h_x - m_x], [0, h_y - m_y]),$$
$$([l_x - m_x, 0], [l_y - m_y, 0]), \quad ([0, h_x - m_x], [l_y - m_y, 0]).$$

実際に ifplot で試したところ，全描画領域（標準では 300 × 300 ドット）に対し sign char. func. を用いるのと同程度まで効率が改善することを観測した（用いた曲線はハート型 H, スペード型 S, ... 等）．

初期ブロック化

描画する場合，全空間が真っ白ということは普通はない．よって，全空間に対して零点が存在しないことを確かめるための評価を行うことは無駄である．また，1 ライン当たり数ドットは描画されるのが普通なので，判定は最初からいくつかのブロックに分割して始めるのが妥当であろう．

ブロックの矩形領域の大きさは，上述のシフトを考慮すれば，2 の冪乗にすべきである（例えば，$300 = 4 \times 64 + 32 + 8 + 4$）．同様の理由により，ブロックは正方形にするのが良いであろう（実際には，その効果は零点の分布に依る）．

6 有理区間数

前にも述べたとおり，区間数は元来（正確な値 ± 誤差）を表すためのもので，その場合は両端の値 $[l, h]$ の正確さは求められない．よって，両端の値は適当な精度をもった浮動小数で表せば実用上は十分である．一方，我々の利用法では，両端の値によって区切られた区間領域（セル）が連続的に接続して描画領域全体を覆いつくすことを仮定している．しかし，ある境界の値をその両側の区間の上限値・下限値として指標関数を浮動小数で計算した場合，区間の連続性が計算結果において保証されているかは定かではない．例えば単調な関数（$f(x)$）をある点において浮動小数で評価する場合，丸めの方向によって結果に違い（$X_1 \neq X_2$）が生ずる可能性があり，その違いに対応する区間（両端は $f^{-1}(X_1)$ と $f^{-1}(X_2)$）が存在することになる．我々の指標関数においてもこのような区間が生じうることは否定できない．このことは実用上は殆ど問題になることはないだろうが，あくまでも正確な計算を行う数式処理において描画でも正確さを極めるためには，何らかの計算手法を用意する必要がある．我々は，この目的のために，区間数の両端の値を有理数で正確に表す有理区間数を提案し，整係数多項式に対する指標関数の有理区間数に対する効率のよい計算法を検討する．

有理区間数と指標関数の計算

整数係数の多項式[2]を区間指標関数の評価に基づいて描画する場合，区間の両端を有理数で表し，式の評価を数式処理的に正確に行えば，上記のような心配なしに完全に正確な評価が可能である．判定をするために計算結果として必要なのは多項式の値の上限と下限の値の符号だけなので，コストのかかる有理数計算によらずとも通分した上で整数計算を行えば十分である．

[2] 係数や式全体が有利的な場合も通分してしまえばよい．

一方，効率の面では全描画空間に対し区間指標を用いると計算量が膨大なため，マルチコアCPUが一般化した今日の計算環境においては並列処理は不可欠であることは実験結果に示したとおりだが，有理区間数を用いて無限の精度で計算する場合は尚更である．しかし，データ表現に要するメモリ量が変化する多倍長数の演算はメモリ管理が必要となり並列処理には向かない．これを解決する一般的な方法がChinese remainderingである（中国剰余定理に基づき，十分に多くの素数のもとでの剰余で計算を行い，正確な計算と対応づける）．以降では，有理区間数をChinese remainderingした場合に，区間数の計算が成り立つか，更に，指標関数の判定が可能かを検討する．

ここまでの議論をまとめる．

- 描画という用途の場合，セルの端点である格子点は有理数で表現し，多項式の評価は（通分の後）すべて整数値で正確に行えば良い
- 並列処理の利用は不可欠．それにはメモリ管理は避けたい，よって，通常の多倍長計算は避けたい．
- 全領域もしくはブロック化した各四分木の根に対応する矩形領域に対し，一度は正確な計算を行い，とりうる値の絶対値の上限（Mとする）を確定する．
- 区間数の上下限値をChinese remainer化した時，区間数としての演算は可能か？主たる計算は多項式の評価で，区間数の演算は加減乗算と冪乗である．

Chinese remainder化された有理区間数の演算

扱う整数の絶対値の上限をMとし，用いる法（素数）$m_1,...,m_r$は$2M \leq m_1 \cdots m_r$とする．任意の整数（但し，$|U| \leq M$）を，その剰余$u_j = U \bmod m_j$の組で表す．但し，$2|u_j| \leq m_j$とする．剰余数の算術演算は通常どおりだが，その区間数の場合は上下限の値の条件により計算式が異なる．区間数の算術演算は既に示したとおりであり，Chinese remainder化した整数の区間数の計算を進めるには，算術演算そのもの以外に次の2つの条件判定が必要となる．

(a) 符号判定，

(b) 2つの値の大小比較．

剰余の組$(u_1,...,u_r)$からU（一般には多倍長）を求めずにこれらの条件判定が可能だろうか？剰余の組からUを復帰するためのガーナー算法では，$U = v_1 + m_1(v_2 + m_2(\cdots(v_{r-1} + m_{r-1}v_r)\cdots))$（ここでも$2|v_j| \leq m_j$とする）を満たす$v_i$を順次求めるが，これを$U$の混合基数表現（mixed-radix representation）と見なせば値の組$(v_1,...,v_r)$だけで条件判定が可能である．即ち，

(a) 【符号判定】$(v_1,...,v_r)$が表す整数の符号：$v_{j+1} = \cdots = v_r = 0$の時$v_j$の符号に等しい

(b) 【大小比較】$(v_{11},...,v_{1r}) \gtrless (v_{21},...,v_{2r})$の大小比較：$i = j+1,...,r$に対し$v_{1i} = v_{2i}$の時$v_{1j} \gtrless v_{2j}$

混合基数表現を求めるガーナーの算法は次のとおり．

- $v_1 \Leftarrow u_1$.
- $k = 2,...,r$の順に：（$v_1,...,v_{k-1}$は計算済みとして）先ず$v_k^{(1)} \Leftarrow u_k$とおき，

$$v_k^{(i+1)} \Leftarrow (v_k^{(i)} - v_i)(m_i^{-1} \bmod m_k) \bmod m_k$$

を$i = 1,...,k-1$に対し繰返し，$v_k \Leftarrow v_k^{(k)}$を得る．

これらの計算は高々倍精度の演算と決まった量のメモリ領域があれば可能である．よって，並列処理化することも容易である．

7 まとめ

本稿では，Risa/Asir の陰関数描画機能 ifplot で用いるアルゴリズムの内，区間指標関数による方法を改良する方法の検討を行った．具体的には，区間数での多項式の評価法について，我々が従来から抱いていた経験則や関連する事実を検討し，より精度良く計算するための方策を提案した．また，これらについては，これまでに試験的に行ったマルチコア向けのマルチスレッド並列処理がほぼそのまま適用可能であることを確認している．また，四分木アルゴリズムについても現実的な観点から効率の改善をするための方法を提案した．更に，数式処理の観点からは描画においても絶対的な正確さを追求する方法が提供されるべきであると考え，その効率的な方法を提案しその並列処理の可能性も示した．

今後，以上を踏まえた上で，実証と開発を進めることが課題となる．具体的には，区間数評価用の基本演算ライブラリと開発，そのマルチスレッド並列処理機構，ブロック化したアルゴリズムと粒度が不均一なタスクの並列処理，Chinese remainder 化した有理区間数の実装（並列処理は必須）などである．更に，区間数の列に対するベクトル処理/SIMD 処理も検討していく必要がある．

参 考 文 献

[1] Ramon E. Moore. *Methods and Applications of Interval Analysis.* SIAM Studies in Applied Mathematics. 1979.

[2] T. Saito, Y. Kondoh, Y. Miyoshi, and T. Takeshima. Faithful plotting of real curves defined by bivariate rational polynomials. 情報処理学会論文誌, Vol. 41, No. 4, pp. 1009–1017, 2000.

[3] 近藤祐史, 村尾裕一, 齋藤友克. Risa/Asir の ifplot の改良と並列化の試み. 京都大学数理解析研究所講究録, No. 1568, pp. 185–191, 2007.

[4] 近藤祐史, 村尾裕一, 齋藤友克. 数式の零点描画の高速化. 数式処理, Vol. 14, No. 2, pp. 27–31, 2007.

[5] 齋藤友克, 近藤祐史, 三好善彦, 竹島卓. Displaying real solution of mathematical equations. *J.JSSAC*, Vol. 6, No. 2, pp. 2–21, 1998.

[6] 平野照比古. 区間数による多項式の評価について. 京都大学数理解析研究所講究録, No. 1295, pp. 220–223, 2002.